Augmented Reality Technician Assistance Program

Jackson W. Volante  
*Cedarville University*, jvolante@cedarville.edu

Kidron R. Filbrun  
*Cedarville University*, kfilbrun@cedarville.edu

Jeffrey T. Deane  
*Cedarville University*, jdeane@cedarville.edu

Andrew T. Carr  
*Cedarville University*, atcarr@cedarville.edu

Nathan A. Harold  
*Cedarville University*, nharold@cedarville.edu

See next page for additional authors
Presenters
Jackson W. Volante, Kidron R. Filbrun, Jeffrey T. Deane, Andrew T. Carr, Nathan A. Harold, and Daniel J. Eckley

This poster is available at DigitalCommons@Cedarville: http://digitalcommons.cedarville.edu/research_scholarship_symposium/2017/poster_presentations/14
Augmented Reality Technician Assistance Program

Abstract

THE AUGMENTED REALITY TECHNICIAN ASSISTANCE PROGRAM is a proof-of-concept project for allowing a remote expert to both communicate with and assist a field technician in completing procedures with which the technician may be unfamiliar. For example, an expert in Navy aircraft maintenance could advise an Air Force flight mechanic about performing repairs or maintenance on Navy aircraft. The end goal of our Cedarville computer science capstone project is to create an experimental prototype to deliver to the Air Force Research Laboratory. In our prototype the expert uses a Microsoft Surface Pro tablet to communicate via marked up still images with a HoloLens-enabled field technician. As a wearable computer, the HoloLens provides the technician a hands-free advantage over traditional devices, because it’s interface does not use typical input devices such as a mouse and keyboard. The advantage of the Surface Pro is that it allows the expert to markup instructional images naturally with the stylus, providing better precision for the annotations.

SOFTWARE DESIGN AND DEVELOPMENT PROCESS

• Background Research – Assess systems’ capabilities and learn how similar problems were solved.
• Requirements Analysis – Discuss desired capabilities with the application's client.
• System Design – Decompose application into components for implementation allowing task delegation to individuals while ensuring the system remains cohesive.
• Implementation – Develop code to perform the functions of the system.
• Test – Verify the system operates properly, fixing any issues that are found.
• Iterate – Cyclically repeat this process throughout the development lifecycle of the product to incrementally improve its capabilities.

SOFTWARE DIAGRAMING – UNDERSTANDING THE RELATIONSHIPS IN CODE

Diagrams can help programmers to visualize the relationships between parts of a project. It is important to understand the project as a whole to track down errors, and avoid duplicating functionality in different components of a large project.

Applications of Software Diagramming

To the Right: A diagram illustrating the most basic kinds of data exchanged between both halves of the application and the users of the application. These very basic drawings are used early in the design process to understand what is expected of the application.

To the Left: A diagram illustrating a more detailed exchange of data. Diagrams like this are useful in documentation, and in fleshing out the details of an implementation before it is actually turned into code.